PRICE $ 2.00

THE 80 NOTEBOOK

AUGUST 1980 ISSUE #5

NOTE: The term "TRS-80" is a registered trademark of Radio Shack, a Division of Tandy Corporation. THE 80 NOTEBOOK is not affiliated with Radio Shack or Tandy Corporation in any way.

"WHAT IS DATA BASE MANAGEMENT AND HOW CAN YOU MAKE USE OF IT?"

The term data base management refers to the creation, maintenance and inquiry to information collectively stored in a file. This file could be located on cassette tape or on a floppy disk.

For the purposes of this article, we will be considering that our data base file is located on cassette tape so that a normal 16K Level 2 BASIC system could use the concepts to be explored.

For those of you with little computer experience, let's look at some rudimentary terminology concerning our subject:

Think of a file as a filing cabinet in an office. If we examine the drawers of that filing cabinet, we will find records. Each record in the file would be like a folder in the drawers of the filing cabinet. Each record contains one or more pieces of information called fields. Each field would be like a document in an individual folder.

Now let's look at an example of what we are discussing:

In our example, we have a file of inventory items to be sold by a store. Each record in a file like this would represent one of the items sold by the store.

The fields we might expect in each record would be - an item number (an identification number or catalog number perhaps), item description text, selling price, number of items on hand, and so on.

Any and all information needed about an item must be represented by fields in each record. Even if only certain items in our file required a certain field, every record in the file must contain that field. The arrangement, number and type of fields in each record must be the same - a uniform file format.

This is only one example of what a data base file might contain. The uses of such a file are endless. Any collection of items or people on which statistics may be kept can be maintained in a data base file.

Now let's look at the kind of program needed to create, maintain and examine a data base file of this type:

Such a program must have several functions which could be selected by a menu routine. This routine would display the names of these functions along with a selection code and then ask the user which function he desires.

Once this selection has been made, the routine would pass control to the appropriate function routine.

These functional routines would perform the following tasks: 1) initialize a file, 2) read a data base from tape, 3) find a record, 4) display a record, 5) change a record, 6) reformat a file, 7) sort a file, 8) write a data base to tape, and 9) report a file on the printer.

Here is the BASIC code which would carry out this selection process:

```
1ØØ PRINT "SELECTION MENU"
11Ø PRINT "1) INITIALIZE A FILE"
12Ø PRINT "2) READ A DATA BASE FROM TAPE"
13Ø PRINT "3) FIND A RECORD"
14Ø PRINT "4) DISPLAY A RECORD"
15Ø PRINT "5) CHANGE A RECORD"
16Ø PRINT "6) REFORMAT A FILE"
17Ø PRINT "7) SORT A FILE"
18Ø PRINT "8) WRITE A DATA BASE TO TAPE"
19Ø PRINT "9) REPORT A FILE ON THE PRINTER"
2ØØ PRINT "1Ø) END": PRINT
21Ø INPUT "PLEASE ENTER YOUR SELECTION"; I
22Ø ON I GOTO 1ØØØ, 2ØØØ, 3ØØØ, 4ØØØ, 5ØØØ, 6ØØØ, 7ØØØ, 8ØØØ, 9ØØØ
23Ø END
```

Now let's take a look at these functions individually in more detail:

The initialization of a file is required to use or create any file under control of this system. Whenever a file is manipulated by this system, it is done so from a table in memory where the file is stored during the manipulation. This table has an entry for each record in the file (or potential or future record in that file).

Before a file could be read into this table in memory or created for the first time, the system must be told the maximum number of records the file will or does contain. This number must allow for the necessary record room for any future record adds to an existing file.

In BASIC, tape records have a maximum allowable record length. Because of this, our data base system will allow a maximum record length of 250 characters.

Once the system knows how many records are to be allocated to the current file, the routine will set up a table of N number of 250 character blank records.

Each blank record in the table denotes an unused record position in the file which can be filled by either a tape record read or a record change function.

When a file is eventually written to tape by the appropriate function, any unused record positions will be treated as deleted records and will not be written to tape.

Here is the BASIC code which would handle this function:

```
1ØØØ INPUT "HOW MANY RECORDS ARE IN THIS FILE"; K
1Ø1Ø J=(K*27Ø)+1ØØØ
1Ø2Ø CLEAR J
1Ø3Ø J=FRE(B$)
1Ø4Ø N=(J-1ØØØ)/27Ø
1Ø5Ø DIM A$(N)
1Ø6Ø FOR I=Ø TO N
1Ø7Ø A$(I)=STRING$(25Ø," ")
1Ø8Ø NEXT I
119Ø GOTO 1ØØ
```

Note the use of the FRE function in this routine to retrieve the total number of available string space characters. Once the string space is computed from the number of records in the file and the string space is "CLEAR"-ed, the values in K and J are lost.

Because of this, the FRE function is used to get back the J value, and from that, the number of records in the file can easily be computed so that the DIM statement can be executed to set up the file table requested.

Once the file table has been initialized, if a previously created data base file is to be manipulated, it must be read into the file table. This can be handled by the second function in our data base program.

Should the number of records on the tape exceed the file table size specified in the initialization function, the system must abort the tape read operation. If this does happen, you must re-execute the initialization function before continuing.

Here is the BASIC code which would read a data base file from a tape into the memory table:

```
2ØØØ K=Ø
2Ø1Ø INPUT #-1,B$
2Ø2Ø IF B$="/*" THEN 1ØØ
2Ø3Ø K=K+1
2Ø4Ø IF K.GT.N THEN 2Ø8Ø
2Ø5Ø IF LEN(B$).NE.25Ø THEN 211Ø
2Ø6Ø A$(K)=B$
2Ø7Ø GOTO 2Ø1Ø
2Ø8Ø PRINT "YOU HAVE EXCEEDED YOUR FILE TABLE SPACE"
2Ø9Ø PRINT "YOU MUST RE-INITIALIZE THE FILE TABLE SPACE"
21ØØ GOTO 1ØØØ
211Ø PRINT "RECORD LENGTH IS NOT 25Ø"
212Ø PRINT "THIS CAN NOT BE A DATA BASE TAPE FILE"
213Ø IF K.GT.1 THEN 2Ø9Ø
214Ø GOTO 1ØØ
```

Note the use of a "/*" record as the last or end of file record on our data base file tape format. When this record is sensed, the routine returns to the function menu routine.

Also, the routine checks the record length to insure that the tape being read is a data base file tape. If a record length other than 250 occurs, the routine will request that the file table initialization function be executed if the condition occurs on other than the first record of the tape file read operation.

After records have been established in the file table, you may wish to locate a specific record in the file using a predefined search argument. Since your quest for a record may require the examination of several fields, this function must set up a table of field positions and search values.

Since the system should set some limit on the number of fields to be examined in the search, this routine will allow up to 20 fields to be specified in the search.

The following is the BASIC code needed to set up the field definition tables to be used in the record search function and in other functions to be discussed later:

```
1Ø9Ø DIM X(2Ø),Y(2Ø),V$(2Ø),Z(2Ø)
11ØØ FOR I=Ø TO 2Ø
111Ø X(I)=Ø
112Ø Y(I)=Ø
113Ø V$(I)=" "
114Ø Z(I)=Ø
115Ø NEXT I
```

With this in mind, let's take a look at how this field definition table system works. The X table holds the starting position of the field in the record. Therefore, this value would be in the range 1 to 250. The Y table holds the size of the field being defined. This value must be from 1 to 250.

The result of the starting position in X + the field size in Y minus 1 must never exceed 250 because this would indicate a field size going outside of the record.

The Z table indicates if a field is numeric by a value of 1. Numeric fields are stored in a record in their string form.

Now let's look at the BASIC code needed to initialize the field definition table with the desired search arguments:

```
3ØØØ K=Ø
3ØØ5 S=Ø: T=Ø: R=Ø: B$=" "
3Ø1Ø INPUT "FIELD STARTING POSITION FROM 1 TO 250"; S
3Ø2Ø IF S=Ø THEN 33ØØ
3Ø3Ø IF S.LT.Ø OR S.GT.25Ø THEN 3Ø1Ø
3Ø4Ø INPUT "FIELD SIZE"; T
3Ø5Ø IF T.LT.1 OR T.GT.25Ø OR S+T-1.GT.25Ø THEN 3Ø4Ø
```

```
3Ø6Ø INPUT "ENTER Ø FOR TEXT FIELD, 1 FOR NUMERIC FIELD"; R
3Ø7Ø IF R.LT.Ø OR R.GT.1 THEN 3Ø6Ø
3Ø8Ø K=K+1
3Ø9Ø X(K)=S
31ØØ Y(K)=T
311Ø Z(K)=R
3115 IF R=1 THEN 314Ø
312Ø INPUT "VALUE TO BE SEARCHED FOR"; B$
313Ø GOTO 316Ø
314Ø INPUT "VALUE TO BE SEARCHED FOR"; H
315Ø B$=STR$(H)
316Ø G=LEN(B$)
317Ø IF G.GT.T OR G.LT.1 THEN 3115
3175 IF G.LT.T THEN B$=B$+STRING$(T-G," ")
318Ø V$(K)=B$
319Ø IF K.LT.2Ø THEN 3ØØ5
32ØØ GOTO 33ØØ
```

Note that in the above routine, the field search value must be within the field size constraint. Smaller text field values are padded with blanks.

Now that we have our search arguments, we can take a look at how the search is carried out. To make the search as flexible as possible, the routine will ask for a starting record number from 1 to N, where N is the file table size.

This will allow the user to search for more than one record with the desired search arguments by repeating a search starting at the next highest record number from the last record matched in such a search.

Here is the BASIC code needed to carry out this search function:

```
33ØØ INPUT "STARTING RECORD NUMBER"; J
331Ø IF J=Ø THEN 1ØØ
332Ø IF J.LT.Ø OR J.GT.N THEN 33ØØ
333Ø FOR I=J TO N
3335 F$=A$(I)
334Ø FOR M=1 TO K
335Ø S=X(M)
336Ø T=Y(M)
337Ø R=Z(M)
338Ø B$=V$(M)
339Ø IF R=1 THEN H=VAL(B$): GOTO 341Ø
34ØØ IF B$.NE.MID$(F$,S,T) THEN 346Ø ELSE 343Ø
341Ø P$=MID$(F$,S,T)
342Ø IF H.NE.VAL(P$) THEN 346Ø
343Ø NEXT M
344Ø PRINT "MATCH AT RECORD NUMBER"; I
345Ø GOTO 33ØØ
346Ø NEXT I
347Ø PRINT "NO MATCH"
348Ø GOTO 1ØØ
```

Notice that a response of zero to the starting record number prompt will end the search mode and return the system to the menu routine.

Once a matching record has been found, that record's record number or position in the data base file table is printed. Note the use of the VAL function to retrieve the numbers from numeric string fields and numeric search arguments to be used in making comparisons.

Once a record has been located and its record number is known, the user will want to display that record on the screen. The display function will ask for the desired record's location number and will display the record along with a scale to allow the user to easily find individual fields on a position by position basis.

Here is the BASIC code which will display a record:

```
4ØØØ INPUT "DESIRED RECORD NUMBER"; J
4ØØ5 IF J.LT.Ø OR J.GT.N THEN 4ØØØ
4Ø1Ø IF J=Ø THEN 1ØØ
4Ø2Ø PRINT STRING$(9,"Ø"); STRING$(1Ø,"1"); STRING$(1Ø,"2"); STRING$(1Ø,"3")
     ; STRING$(1Ø,"4"); STRING$(1Ø,"5"); STRING$(4,"6")
4Ø25 FOR I=1 TO 6
4Ø3Ø PRINT "1234567890";
4Ø35 NEXT I
4Ø4Ø PRINT "123"
4Ø5Ø F$=A$(J)
4Ø6Ø PRINT MID$(F$,1,63)
4Ø7Ø PRINT STRING$(36,"Ø"); STRING$(27,"1")
4Ø8Ø PRINT STRING$(6,"6"); STRING$(1Ø,"7"); STRING$(1Ø,"8"); STRING$(1Ø,"9")
     ; STRING$(1Ø,"Ø"); STRING$(1Ø,"1"); STRING$(7,"2")
4Ø9Ø FOR I=1 TO 6
41ØØ PRINT "4567890123";
411Ø NEXT I
412Ø PRINT "456"
413Ø PRINT MID$(F$,64,63)
414Ø PRINT STRING$(63,"1")
415Ø PRINT STRING$(3,"2"); STRING$(1Ø,"3"); STRING$(1Ø,"4"); STRING$(1Ø,"5")
     ; STRING$(1Ø,"6"); STRING$(1Ø,"7"); STRING$(1Ø,"8")
416Ø FOR I=1 TO 6
417Ø PRINT "7890123456";
418Ø NEXT I
419Ø PRINT "789"
42ØØ PRINT MID$(F$,127,63)
421Ø PRINT STRING$(1Ø,"1"); STRING$(51,"2")
422Ø PRINT STRING$(1Ø,"9"); STRING$(1Ø,"Ø"); STRING$(1Ø,"1"); STRING$(1Ø,"2"
     ); STRING$(1Ø,"3"); STRING$(1Ø,"4"); "5"
423Ø FOR I=1 TO 6
424Ø PRINT "0123456789"
425Ø NEXT I
426Ø PRINT "Ø"
427Ø PRINT MID$(F$,190,61)
428Ø PRINT "HIT THE X KEY TO CONTINUE";
429Ø B$=" "
43ØØ B$=INKEY$
431Ø IF B$.NE."X" THEN 429Ø
432Ø PRINT " "
433Ø GOTO 4ØØØ
```

Note that the desired record number prompt will allow the user to enter zero to return to the selection menu routine.

When the display of the currently selected record is completed, the user must hit the X key to allow the system to continue. The X key will return the user to the record number prompt to allow the user to display another record.

Once we have examined a record through the display function, some of the information in that record may require some changes. To do this, the change function allows the user to modify fields in a record by specifying the field starting position, field size and the new value for that field. This activity can be continued for as many field changes as desired in a record and for as many records requiring changes before returning to the menu routine.

Here is the BASIC code needed to perform these field changes:

```
5ØØØ INPUT "DESIRED RECORD NUMBER"; J
5Ø1Ø IF J=Ø THEN 1ØØ
5Ø2Ø IF J.LT.Ø OR J.GT.N THEN 5ØØØ
5Ø3Ø F$=A$(J)
5Ø4Ø INPUT "FIELD STARTING POSITION"; S
5Ø5Ø IFS=Ø THEN A$(J)=F$: GOTO 5ØØØ
5Ø6Ø IF S.LT.Ø OR S.GT.25Ø THEN 5Ø4Ø
5Ø7Ø INPUT "FIELD SIZE"; T
5Ø8Ø IF T.LT.1 OR T.GT.25Ø OR T+S-1.GT.25Ø THEN 5Ø7Ø
5Ø9Ø PRINT "FIELD WAS"; MID$(F$,S,T)
51ØØ INPUT "NEW FIELD VALUE"; B$
511Ø H=LEN(B$)
5115 IF H.LT.1 OR H.GT.T THEN 51ØØ
512Ø IF H.LT.T THEN B$=B$+STRING$(T-H," ")
513Ø P$=""
514Ø IF S.GT.1 THEN P$=MID$(F$,1,S-1)
515Ø P$=P$+B$
516Ø IF S+T-1.LT.25Ø THEN P$=P$+MID$(F$,S+T,25Ø-(S+T-1))
517Ø F$=P$
518Ø GOTO 5Ø4Ø
```

When entering the desired record number, if the user enters a zero, the change routine will return to the menu routine.

If the user enters a zero to the field starting position prompt, the change routine will allow the user to return to the desired record number prompt to select another record to be changed.

Since the user is specifying the starting field positions and field sizes, it is important that you plan the field layout of your records carefully so that there are no overlapping fields in your record layout. Remember - if you are going to search, sort, reformat or report your file, every record must have the same record layout of fields!

When dealing with numeric fields, you must plan the field sizes to be large enough to hold the string form of a number along with its leading blank, sign character, decimal point, maximum number of digits possible in each field, and possibly the exponent notations characteristic of a floating point number if you are using such numeric information.

When you wish to insert a new record in the file, use the find function to locate a blank record - an unused record position to be used for your new record. Be sure that as you fill the fields of a new record with information that at least one of those fields will always contain some non-blank text regardless of any of the possible data changes that may occur in the future of that record so that the record will never be completely filled with blanks and considered an unused record position.

This can be done by including a record identification field when planning your record layout. This field would be a one character field containing a non-blank character which is never altered unless the record is deleted.

To delete a record from the file, simply fill the entire record with blanks. The easiest way to do this is to enter a field change treating the record as one 250 character field and entering a blank for the new field value.

When entering new field values, it is not necessary to input a value with exactly the same number of characters as the field it is for. The value length must not exceed the field size but if it is less, the system will pad the field to the right with blanks.

When you wish to blank out a field, you need only enter a single blank as the new value because of this.

In the course of maintaining a data base file, a time may come when your original record field layout becomes inadequate for one reason or another. This may happen because of the need to sandwich a new field between two existing fields in a file layout.

To do this, the reformat function allows the user to rearrange the layout of fields in each record allowing blank areas where desired.

The first operation this function will carry out is to determine how to move the existing fields around into the new record format. For this, the user must input each field's starting position in the current record format, the field size and the starting position in the new record layout.

Once this is known, the actual reformatting can be carried out on the file table records. Since this operation uses the X, Y and Z tables discussed earlier, the maximum number of field redefinitions is limited to 20.

Here is the BASIC code needed for this function:

```
6ØØØ K=Ø
6Ø1Ø INPUT "FIELD STARTING POSITION"; S
6Ø2Ø IF S=Ø THEN 63ØØ
6Ø3Ø IF S.LT.Ø OR S.GT.25Ø THEN 6Ø1Ø
6Ø4Ø INPUT "FIELD SIZE"; T
6Ø5Ø IF T.LT.1 OR T.GT.25Ø OR T+S-1.GT.25Ø THEN 6Ø4Ø
6Ø6Ø INPUT "NEW STARTING POSITION"; R
6Ø7Ø IF R.LT.1 OR R.GT.25Ø OR R+T-1.GT.25Ø THEN 6Ø6Ø
6Ø8Ø K=K+1
6Ø9Ø X(K)=S
```

```
61ØØ Y(K)=T
611Ø Z(K)=R
612Ø IF K.LT.2Ø THEN 6Ø1Ø
63ØØ FOR I=1 TO N
631Ø P$=STRING$(250," ")
632Ø F$=A$(I)
633Ø FOR J=1 TO K
634Ø S=X(J)
635Ø T=Y(J)
636Ø R=Z(J)
637Ø B$=""
638Ø IF R.GT.1 THEN B$=MID$(P$,1,R-1)
639Ø B$=B$+MID$(F$,S,T)
64ØØ IF R+T-1.LT.25Ø THEN B$=B$+MID$(P$,R+T,25Ø-(R+T-1))
641Ø P$=B$
642Ø NEXT J
643Ø A$(I)=P$
644Ø NEXT I
645Ø GOTO 1ØØ
```

Again, when the user enters zero to the field starting position prompt, this signals the user has finished entering the field redefinitions for his new file record format with less than 2Ø field redefinitions in total.

This function should be preceeded by the file initialization function followed by data base file read function. After the reformat function is completed, the file should be written to tape by the data base file tape write function.

In the course of displaying your data base file records, before using the report file function and possibly before writing a data base file to tape or after a reformat function, you may wish to sort your file into a particular sequence. This can be done using the sort function.

To do this, using the X, Y and Z tables, the user will specify the fields on which the file is to be sorted by entering the starting position, field size and sort type of each sort field required up to a maximum of 20 fields.

The sort types allowed by the routine are 1) ascending text field, 2) descending text field, 3) ascending numeric field and 4) descending numeric field.

Here is the BASIC code needed to input the sort arguments and perform the actual sort. As before, entering a zero to the field starting position prompt will signal the end of the sort argument input phase.

```
7ØØØ K=Ø
7Ø1Ø INPUT "FIELD STARTING POSITION"; S
7Ø2Ø IF S=Ø THEN 73ØØ
7Ø3Ø IF S.LT.Ø OR S.GT.25Ø THEN 7Ø1Ø
7Ø4Ø INPUT "FIELD SIZE"; T
7Ø5Ø IF T.LT.1 OR T.GT.25Ø OR S+T-1.GT.25Ø THEN 7Ø4Ø
7Ø6Ø INPUT "SORT TYPE"; R
7Ø7Ø IF R.LT.1 OR R.GT.4 THEN 7Ø6Ø
7Ø8Ø K=K+1
```

```
7Ø9Ø X(K)=S
71ØØ Y(K)=T
711Ø Z(K)=R
712Ø IF K.LT.2Ø THEN 7Ø1Ø
73ØØ FOR J=1 TO K
731Ø S=X(J)
732Ø T=Y(J)
733Ø R=Z(J)
7335 L=Ø
734Ø FOR I=1 TO N
735Ø B$=A$(I)
736Ø F$=A$(1): M=1
737Ø IF I.LT.N THEN B$=A$(I+1): M=I+1: F$=A$(I)
738Ø ON R GOTO 74ØØ, 75ØØ, 76ØØ, 77ØØ
74ØØ IF MID$(F$,S,T).GT.MID$(B$,S,T) THEN P$=A$(I): A$(I)=A$(M): A$(M)=P$:
     L=1
741Ø GOTO 78ØØ
75ØØ IF MID$(F$,S,T).LT.MID$(B$,S,T) THEN P$=A$(I): A$(I)=A$(M): A$(M)=P$:
     L=1
751Ø GOTO 78ØØ
76ØØ P$=MID$(F$,S,T): G=VAL(P$)
761Ø P$=MID$(B$,S,T): H=VAL(P$)
762Ø IF G.GT.H THEN P$=A$(I): A$(I)=A$(M): A$(M)=P$: L=1
763Ø GOTO 78ØØ
77ØØ P$=MID$(F$,S,T): G=VAL(P$)
771Ø P$=MID$(B$,S,T): H=VAL(P$)
772Ø IF G.LT.H THEN P$=A$(I): A$(I)=A$(M): A$(M)=P$: L=1
78ØØ NEXT I
781Ø IF L=1 THEN 7335
782Ø NEXT J
```

Remember - blank (deleted or unused) records are also sorted into order!

When you are entering sort field arguments, be sure you enter them in lowest to highest order of sorting priority. For example, if sorting on two fields, the first field defined for the sort would be the minor sort field and the second field defined for the sort would be the major sort field.

After a file has been changed, reformatted or sorted, it should be saved in its updated form on tape. This can be done by using the data base file tape write function. This function will drop all deleted records (unused, blank records) from the file as it is copied to tape.

Here is the BASIC code needed to perform this function:

```
8ØØØ FOR I=1 TO N
8Ø1Ø IF A$(I)=" " THEN 8Ø3Ø
8Ø2Ø PRINT #-1, A$(I)
8Ø3Ø NEXT I
8Ø4Ø PRINT #-1, "/*"
```

We have one more function available to the user in our system - the file report function. Once a file has been initialized, read and optionally sorted, it can be listed on a printer in a formatted fashion.

This is done by giving the user an 80 character, blank print line to work with and allows him to select what fields from the file records he wishes to print and where in the print line they should be placed. Blank file records are ignored in this printing of the file.

Using the X, Y and Z tables, the first section of this routine will allow the user to enter the starting position within a file record, field size and the starting position of the field within the 80 character print line for each field desired to be listed up to a maximum of 20 fields.

Again, when the user enters a zero to the starting position within a file record prompt, this will signal that the user has finished entering his field selection definitions and the routine can perform the actual listing operation.

Here is the BASIC code needed to perform this function:

```
9ØØØ K=Ø
9Ø1Ø INPUT "ENTER FIELD STARTING POSITION WITHIN RECORD"; S
9Ø2Ø IF S=Ø THEN 93ØØ
9Ø3Ø IF S.LT.Ø OR S.GT.25Ø THEN 9Ø1Ø
9Ø4Ø INPUT "FIELD SIZE"; T
9Ø5Ø IF T.LT.1 OR T.GT.25Ø OR T+S-1.GT.25Ø THEN 9Ø4Ø
9Ø6Ø INPUT "FIELD STARTING POSITION WITHIN PRINT LINE"; R
9Ø7Ø IF R.LT.1 OR R.GT.8Ø OR R+T-1.GT.8Ø THEN 9Ø6Ø
9Ø8Ø K=K+1
9Ø9Ø X(K)=S
91ØØ Y(K)=T
911Ø Z(K)=R
912Ø IF K.LT.2Ø THEN 9Ø1Ø
93ØØ FOR I=1 TO N
931Ø F$=A$(I)
932Ø IF F$=" " THEN 95ØØ
933Ø P$=STRING$(8Ø," ")
934Ø FOR J=1 TO K
935Ø S=X(J)
936Ø T=Y(J)
937Ø R=Z(J)
938Ø B$=""
939Ø IF R.GT.1 THEN B$=MID$(P$,1,R-1)
94ØØ B$=B$+MID$(F$,S,T)
941Ø IF R+T-1.LT.8Ø THEN B$=B$+MID$(P$,R+T,8Ø-(R+T-1))
942Ø P$=B$
943Ø NEXT J
944Ø LPRINT P$
95ØØ NEXT I
951Ø GOTO 1ØØ
```

While this is a simplified type of data base management system for the TRS-80, it does provide the most needed capabilities while using the least hardware and software resources.

Some of the suggestions we would have for possible improvements include: headings, totals and group breaks on reports, adding the capability to the search function to select records based on a range of values for a chosen field, and adding the capability to select records conditionally for acceptance during the

tape read function - this would allow the user to selectively print listings of portions of a current data base file, create sub-files from a larger data base file by writing portions of a data base file to a new tape, selectively create a reformatted sub-file from a larger data base file and selectively sort only the records you desire for a sub-file creation or listing.

If you have 32K or more and like a challenge, you might try devising the BASIC code needed to implement these improvements and any others you might come up with.

Whether you try this or not, we hope you will make valuable use of this compact little data base system and use its techniques in your other programming tasks requiring file manipulation.

THE 80 NOTEBOOK
R.D.#3
Nazareth, PA 18064

BULK RATE
U.S. POSTAGE
PAID
Stockertown, PA 18083
Permit No. 8

4/80

3/81